CCSLM: Applications Beyond Sensor Fusion

Core idea

Concatenated Context-Selected Linear Mappings (CCSLM) provides a simple systems-level description of computation in which context controls a sequence of linear transformations:

$$y = L_n(C_n, x_n) \cdots L_2(C_2, x_2) L_1(C_1, x_1) x$$

- x: input data.
- C_i: context available to stage i.
- $L_i(C_i, x_i)$: the linear mapping selected or modulated by that context.
- The stages are simply **composed in series**.
- Different stages can respond to completely different contextual variables.
- Context is therefore a **control pathway**, separate from the data pathway.

Why this matters

- A conventional viewpoint asks:
- *What function does the network compute?*
- CCSLM asks:
- *What sequence of transformations does the current context cause the system to perform?*
- This turns a seemingly complicated nonlinear system into:
- **context selection**
- followed by **linear computation**
- repeated through a sequence of stages.

1. Sensor fusion

- Natural application because sensors have different reliability in different contexts.
- Context can select or modulate the fusion mapping according to:
- sensor quality
- lighting
- weather
- terrain
- velocity
- GPS availability
- operating mode

- Instead of one fixed fusion function:

  $$y = L(x)$$

  use:

  $$y = L(C, x).$$

- Particularly attractive when the appropriate fusion strategy changes substantially between operating regimes.

2. Multimodal data fusion

• Apply the same principle to:
• vision
• audio
• text
• telemetry
• proprioception
• spatial data
• Context can determine which transformations are appropriate for the current task.
• Examples:
• object identification → emphasize visual features
• navigation → emphasize spatial information
• human interaction → emphasize language/audio
• fault diagnosis → emphasize telemetry
• This provides a **context-dependent transformation of multimodal information**, rather than treating fusion as a single fixed operation.

3. Adaptive and switched control

• Classical control already uses families of controllers:

$$u = K_i x.$$

• CCSLM provides a natural generalization:

$$u = L(C, x).$$

• Context could encode:

• speed
• load
• terrain
• battery state
• operating mode
• proximity to obstacles
• Potentially useful for **hybrid dynamical systems** combining continuous dynamics with discrete operating regimes.

4. Robotics

• Robots naturally possess large amounts of persistent context:
• pose
• velocity
• task
• environment
• sensor status
• battery state
• time
• current behavioural mode

• Different layers/stages can respond to different variables.
• A robot could therefore be viewed as a sequence of context-selected transformations rather than as a single monolithic nonlinear function.
• This provides a natural bridge between:
• neural computation
• control systems
• finite-state machines
• adaptive systems.

5. System identification

• Many physical systems behave approximately linearly within particular operating regimes:

$$y = A_i x.$$

• CCSLM suggests learning an **atlas of local linear mappings** indexed by context.

• Rather than forcing one global nonlinear model to explain every regime:
• identify useful mappings
• identify the contexts in which they apply
• compose them through the system.
• This could provide an interpretable representation of system dynamics.

6. Temporal state and finite-state behaviour

• Context does not have to be instantaneous sensor data.

• It can be persistent state:

$$C_t = f(C_{t-1}, x_t).$$

• The resulting system is:

$$y_t = L(C_t, x_t).$$

• This gives CCSLM a natural connection to:

• finite automata
• state machines
• hysteresis
• temporal multiplexing
• behavioural modes.
• Potentially useful when *what the system should do now depends partly on what happened previously*.

7. Fault tolerance

• Context can encode computational and sensor health.
• The system can select different mappings when:
• a sensor fails

• communication degrades
• computational resources become limited
• battery power falls
• environmental conditions deteriorate.
• This suggests **context-dependent graceful degradation** rather than simply removing failed inputs.

8. Mixture-of-experts

• Conventional mixture-of-experts explicitly combines expert functions.

• CCSLM offers a related but potentially simpler viewpoint:

$$y = L_{C(x)}x.$$

• The "experts" can be regarded as different computational states of a linear mapping.

• Several stages can provide a large number of effective transformations without explicitly constructing a separate complete model for every context combination.
• This may be particularly relevant to the combinatorial-explosion problem.

9. Model compression

• Contexts producing similar mappings need not necessarily have independent parameters.

• Possible representation:

$$L_C = L_0 + \Delta L_C.$$

• Or cluster contexts around shared mappings:

$$C \to k(C) \to L_k.$$

• The **Context Specialization Metric (CSM)** could help identify:

• contexts with effectively identical mappings
• highly specialized contexts
• clusters of related mappings.
• CCSLM therefore potentially provides a route to **structured parameter sharing**.

10. Continual and incremental learning

• New experience need not necessarily modify the entire computational system.
• New context-specific mappings could be learned or existing mappings refined.
• This suggests a separation between:
• learning *what computation is useful*
• learning *when that computation is useful*.
• Potentially attractive for systems operating continuously in changing environments.

11. Goal switching and planning

- Context can include the current goal:

$$C = (\text{goal}, \text{environmental state}).$$

- Changing goals can therefore change the computational regime.

- A blocked goal could cause a transition:

$$C_A \to C_B.$$

- Returning to the original goal can restore its previous computational state.

- This connects CCSLM with:
- behavioural switching
- hierarchical control
- reinforcement learning
- goal-directed robotics.

12. Personalization and adaptation

- Context can represent properties of the current operating environment or user.
- Different contexts can select slightly different transformations without requiring complete retraining.
- This suggests a potentially lightweight mechanism for:
- personalization
- domain adaptation
- environmental adaptation
- machine-specific calibration.

A useful research question

The broad question may be:

> **What classes of computation become easier to represent when context is allowed to select the linear mappings rather than merely being another input to the computation?**

This distinction is important.

- **Conventional approach:** context becomes data.
- **CCSLM approach:** context can remain a **control signal**.
- Data flows through the computational stages.
- Context determines *which transformation the data encounters*.

Potentially important domains

- Sensor fusion

- Multimodal fusion
- Robotics
- Adaptive control
- Hybrid dynamical systems
- System identification
- Fault-tolerant computation
- Continual learning
- Model compression
- Mixture-of-experts
- Temporal/state-dependent computation
- Goal-directed behaviour
- Edge and resource-adaptive AI

The broader hypothesis

CCSLM may be useful wherever a system has the structure:

> **The same kind of data requires different transformations under different circumstances.**

That makes CCSLM potentially less a specialized neural-network architecture and more a **general computational architecture for context-dependent transformation**.

The particularly useful conceptual shift is therefore:

$$\boxed{\text{context does not merely describe the data; context controls the computation}}$$

This may be the central idea worth testing across domains.